Hierarchical conceptual clustering in a first order representation

Ch. Vrain

LIFO - Université d'Orléans - BP 6759
45067 Orléans Cedex 2 - France
email : cv@lifo.univ-orleans.fr

Abstract. This paper presents work in progress on unsupervised learning from observations represented in a subset of first order logic. In most systems, observations are described by pairs (attribute, value) and two main approaches can be distinguished: bottom-up methods that consider all the observations and build a hierarchy of concepts starting from the observations and generalizing them / top-down incremental methods that process observations one after the other and incorporate them by a search down the current hierarchy. Our aim is to test how incremental methods could be adapted to representations written in first order logic and we intend to test several similarity measures and generalization methods in order to study their appropriateness.

1 Introduction

Learning concepts from observations is an important field in Machine Learning that can be divided into two sub-domains:

- supervised learning [9, 13, 6, 14]: positive and negative examples of the concepts are provided to the system and it must learn either characteristic descriptions of the concepts, or discriminant descriptions between them.
- unsupervised learning [10, 4, 8, 3, 2]: only observations are given and the system must determine both the underlying concepts and their descriptions. The terms *concept formation* or *conceptual clustering* are often used to denote such kinds of learning.

Learning concepts from positive and negative examples has been widely used to build knowledge bases for Expert Systems: it seems easier for an expert to provide examples and counterexamples of his behavior rather than giving the usually empirical rules that determine his behavior. An application to air traffic control[1] [7] has shown us that truly an expert can quite easily determine the main concepts of his domain and provide a system with positive and negative examples of them. Nevertheless, these concepts are very often too general and should be decomposed into subconcepts. This task can be difficult and therefore,

[1] contract PERSPICACE, 86 - 88, C.E.N.A., Athis Mons, France / L.R.I., university of Paris 11, France

even in the case of supervised learning, concept formation can be very useful to help an expert structuring his knowledge.

Most of the approaches in the field of concept formation are based on a attribute-value representation of the observations. This formalism is widely used in Machine Learning because it is much more tractable than first order languages. But, in some applications, as for instance in the field of air traffic control [7], this representation language is not expressive enough and we do need a richer formalism. As far as we know, few systems [2, 6] have been developed in first order representations. In this paper, we intend to study how incremental methods, that enable to reduce the complexity and that are more adapted to real applications and/or to knowledge acquisition, could be adapted to first order representations.

The two incremental systems Cobweb [4] and Adeclu [3] process observations represented by pairs (attribute, value). Although their representations of classes differ, they share the same architecture: they are based on a few operations on hierarchies of classes: creating a new class, inserting an observation into a class, merging two classes, splitting a class into its subclasses, ... and the general algorithm to insert a new observation into a hierarchy can be outlined as follows: ($\mathcal{C}$ denotes the current class in which the observation has been inserted):

1. insert the observation into the root of the tree;
   the current class $\mathcal{C}$ is the root of the tree
2. repeat
   choose one of the operations mentioned above;
   update $\mathcal{C}$

until $\mathcal{C}$ contains only this observation.

Algorithm 1

We have developed a hierarchical clustering tool [5] based on the same general algorithm but that handles observations represented in a subset of first order logic. The intensional descriptions of the classes are also described by first order formula. As in the system Adeclu [3], a gain is computed which enables to choose the best operation to perform. The goal of this tool is to test different similarity measures and different ways of computing the intensional description of a class. For the time being, the similarity measure that we have implemented relies on a matching between the objects that compose the two descriptions and the intensional representation of a class is computed by generalization, the generalization process is based on the structural matching principle [6, 14].

2 The classification tool

The tool that we have developed is based on Algorithm 1 given in section 1. But it relies on two functions: a function *gen* that takes as inputs two descriptions and returns a generalization of them and a function *sim* that takes as inputs two descriptions and returns a value representing their similarity. This tool requires that the observations and the classes are described in the same formalism; it

enables to use the same function both for generalizing an observation and a class and for generalizing two classes.

Let us call $\mathcal{R}$ the representation language and let us first precise some definitions:

- In the following, we identify an observation and its description in $\mathcal{R}$. It is clear that if the language is not expressive enough, two distinct observations may have the same representation.
- A class is a set of observations; it is described either extensionally by the set of observations that belong to this class or intensionally. In our logical framework, an intensional description of a class satisfies the two following properties: the intensional description of a class restricted to an observation is the observation and every observation of the class satisfies the intensional description of the class.
- A leaf of the hierarchy corresponds to a unique observation and we identify it with the class composed of this unique observation, [2]

As in the systems Fisher [4] and Adeclu [3], our tool is principally based on the following operations:

- Creating a new concept consists in creating a new class $\mathcal{C}_{n+1}$ composed only of the observation e.
- Inserting an observation e in the class $\mathcal{C}_i$ consists in creating a new class $\mathcal{C}'_i$ composed of e and the elements of $\mathcal{C}_i$.
- Merging two classes $\mathcal{C}_i$ and $\mathcal{C}_j$, $j \neq i$, consists in deleting the classes $\mathcal{C}_i$ and $\mathcal{C}_j$ and replacing them by the class $\mathcal{C}_r$ composed of e, the observations of $\mathcal{C}_i$ and the observations of $\mathcal{C}_j$.
- Splitting a class $\mathcal{C}_i$ consists in replacing the class $\mathcal{C}_i$ by its subclasses $\mathcal{C}_{i_1}$, $\ldots$, $\mathcal{C}_{i_k}$ and inserting e in the best subclasses $\mathcal{C}_{i_j}$.
- Inserting e into an empty hierarchy and inserting e into a class $\mathcal{C}$ which is a leaf.

Each time a class is modified, a new intensional description of the class is computed.

In the system Adeclu [3], a function $sc(e, \mathcal{C}_i, \mathcal{P}_0)$ measures the links between an observation e and a class $\mathcal{C}_i$ of a partition $\mathcal{P}_0$ and the adequation of e to the class $\mathcal{C}_i$ of the partition $\mathcal{P}_0$ is measured by the formula:

$$ad(e, \mathcal{C}_i, \mathcal{P}_0) = sc(e, \mathcal{C}_i, \mathcal{P}_0) - \sum_{\{\mathcal{C}_j \in \mathcal{P}_0, i \neq j\}} sc(e, \mathcal{C}_j, \mathcal{P}_0).$$

The best operation is the operation that maximises the adequation.

Our tool must be provided with a similarity function $sim(D_i, D_j)$ that measures the similarity between two descriptions and a generalization function $gen(D_i, D_j)$ that computes a generalization of two descriptions. If e is an observation and if D_i is the intensional description of the class C_i, then in our tool

[2] We shall see that this assumption is not always desirable and a further improvement of our classification tool will be to relax it.

we take $sc(e, C_i, \mathcal{P}_0) = sim(e, D_i)$. Moreover, the intensional description of the class composed of e and the observations of C_i is $gen(e, C_i)$. In next section, we present the first order representation that we have adopted and we propose two generalization functions and a similarity measure.

3 Application to first order logic

3.1 Representation

Logical representation. The observations and the descriptions of the classes are represented in a subset of first order logic.
Example 1. Let us consider for instance the observations $\mathcal{E}_1$ and $\mathcal{E}_2$, where the first one is composed of a small square put on a large rectangle whereas the second one is composed of a small rectangle put on a large square. The first example may be expressed by the following formulae:

$$(rectangle\ x_1) \wedge (large\ x_1) \wedge (on\ y_1\ x_1) \wedge (small\ y_1) \wedge (square\ y_1) \quad (1)$$

in which *rectangle*, *square*, *large* and *on* are predicate symbols whereas x_1 and y_1 are variables. It means that there exist two objects, represented by the variables x_1 and y_1, x_1 is a large rectangle and y_1 is a square put on x_1. The information that x_1 is a rectangle and y_1 is a square could also have been expressed by *(shape x_1 (rectangle))* and *(shape y_1 (square))*.

The sets of predicate symbols and of constants needed to express the observations must be chosen according to the application. Let us call $\mathcal{P}$ the set of predicate symbols, $\mathcal{C}$ the set of constant symbols and $\mathcal{V}$ the set of variables used to describe the observations and the intensional descriptions of the classes.

Definition 1. *An* atom *is an expression $(p\ t_1 \ldots t_n)$ where $p \in \mathcal{P}$ and for all i, either $t_i \in \mathcal{V}$ or $t_i = (a_i)$ with $a_i \in \mathcal{C}$. An atom $(p\ t_1 \ldots t_n)$ is* ground *if for all i, $t_i \notin \mathcal{V}$, i.e., $t_i = (a_i)$ and $a_i \in \mathcal{C}$. An atom $(p\ t_1 \ldots t_n)$ is* relational *if it contains at least two distinct variables, otherwise the atom $(p\ t_1 \ldots t_n)$ is* unary*. If $\mathcal{A}$ is an atom, we note $Var(\mathcal{A})$ the set of distinct variables of $\mathcal{A}$ and if $\mathcal{D}$ is a conjunction of atoms, we note $Var(\mathcal{D})$ the set of distinct variables that occur in $\mathcal{D}$*

In this work, observations and intensional descriptions are expressed in the same way and the description of a class composed of a single element is equal to the description of this element. The intensional descriptions are computed by generalizing either two observations or an observation and an intensional description of a class. Different ways of generalizing can be considered depending on whether we stress on the relations or on the objects that compose the descriptions. In this latter case, we transform the initial representation into a more "object-oriented" representation, which is equivalent to the first one.

Object representation. For this purpose, we introduce two special symbols $\$x$ and $?y$, where $\$x$ denotes the object under consideration and $?y$ denotes another object.

Definition 2. *A* characteristic *is an atom* $(p\ t_1 \ldots t_n)$ *where* $t_i = \$x$ *or* $t_i = ?y$ *or* $t_i = (a_i)$, $a_i \in \mathcal{C}$. *A characteristic is* unary *if no occurrences of* $?y$ *appears. An object representation,* $\mathcal{D}^{obj}$, *is composed of:*
– *a set, called* $\mathcal{R}el(\mathcal{D}^{obj})$, *of relational atoms,*
– *a set, called* $\mathcal{O}bj(\mathcal{D}^{obj})$, *of pairs* $(x_i, set\_char(x_i, \mathcal{D}^{obj}))$, $i = 1 \ldots n$, *where* x_i *is a variable,* $set\_char(x_i, \mathcal{D}^{obj})$ *is a set of characteristics and* $i \neq j \Rightarrow x_i \neq x_j$.

Example 2. The characteristic $(rectangle\ \$x)$ means that the object under consideration and referred to as $\$x$ is a rectangle; the characteristic $(on\ \$x\ ?y)$ means that it is put on an object referred to as $?y$.

We can associate to a logical representation an object-oriented one, as follows:

Definition 3. *Let* x *be a variable and* $\mathcal{A}$ *an atom in which* x *occurs. The* characteristic *of* x *linked to* $\mathcal{A}$, *written* $char(x, \mathcal{A})$ *is the formula obtained by replacing in* $\mathcal{A}$ *all the occurrences of* x *by the special symbol* $\$x$ *and all the occurrences of variables different from* x *by the special symbol* $?y$. *If* $\mathcal{D}^{log}$ *is a conjunction of atoms and* x *a variable, we note* $set\_char(x, \mathcal{D}^{log}) = \{char(x, \mathcal{A}) |$ $\mathcal{A}$ *belongs to* $\mathcal{D}^{log}$ *and* x *occurs in* $\mathcal{A}\}$.

The set of characteristics linked to an object is important to evaluate the similarities it shares with other objects. This notion was already used in [14].

Example 3. If we consider the representation of observation 1, given by formula 1, then:

$$set\_char(x_1, \mathcal{E}) = \{(rectangle\ \$x)\ \ (large\ \$x)\ \ (on\ ?y\ \$x)\}.$$

Definition 4. *Let* $\mathcal{D}^{log}$ *be a conjunction of atoms. The object-representation linked to* $\mathcal{D}^{log}$ *is defined by:*
– $\mathcal{R}el(\mathcal{D}^{obj}) = \{\mathcal{A} | \mathcal{A}$ *is an element of* $\mathcal{D}^{log}$ *and* $\mathcal{A}$ *is relational*$\}$,
– $\mathcal{O}bj(\mathcal{D}^{obj}) = \{(x, set\_char(x, \mathcal{D}^{log})) | x \in Var(\mathcal{D}^{log})\}$.
It is written $TR_{log \to obj}(\mathcal{D}^{log})$.

For sake of understandability, $\mathcal{D}^{obj}$ represents an object representation of a concept whereas $\mathcal{D}^{log}$ represents a logical representation.

Example 4. The object representation of the example given in formula 1 is translated into:

$$\begin{aligned}
&\mathcal{R}el(\mathcal{D}^{obj}) = \{\ (on\ y_1\ x_1)\}\\
&\mathcal{O}bj(\mathcal{D}^{obj}) = \{\ (x_1\ \ \{(rectangle\ \$x)\ \ (large\ \$x)\ \ (on\ ?y\ \$x)\}\ )\\
&\qquad\qquad\quad (y_1\ \ \{(square\ \$x)\quad (small\ \$x)\ \ (on\ \$x\ ?y)\}\ )\ \ \}
\end{aligned} \tag{2}$$

3.2 Generalization

As has already been mentioned, we can distinguish different ways of generalizing depending on whether we stress on the relations that compose the descriptions or on the objects. We intend to test both kinds of generalization. For the time being, we have implemented a generalization function based on the structural matching principle [6] that gives more importance to the objects (represented by variables) that compose the descriptions than to the relations (expressed by predicate symbols). On the other hand, we are currently implemented a generalization function that computes the least general generalization defined in [12] but we have to solve complexity problems. Let us illustrate both by the following example:

Example 5. Let us suppose that $\mathcal{E}_1$ and $\mathcal{E}_2$ are expressed by:

$$(rectangle\ x_1) \wedge (large\ x_1) \wedge (on\ y_1\ x_1) \wedge (small\ y_1) \wedge (square\ y_1) \quad and \quad (3)$$

$$(square\ x_2) \wedge (large\ x_2) \wedge (on\ y_2\ x_2) \wedge (small\ y_2) \wedge (rectangle\ y_2) \quad (4)$$

(For sake of understandability, in this paper, the variables that appear in a description differ from those that appear in another one. It is not necessary in the current implementation.)

The least general generalization of $\mathcal{E}_1$ and $\mathcal{E}_2$, as defined by Plotkin [12], would be:

$$(large\ x_4) \wedge (on\ y_4\ x_4) \wedge (small\ y_4) \wedge (rectangle\ x_5) \wedge (square\ y_5) \quad (5)$$

In such a generalization, no information is lost, but this formula is quite difficult to understand for an expert, since different variables must be instantiated by the same object to get back the observations. For instance, to prove that the formula 5 is a generalization of $\mathcal{E}_1$, the variables x_5 and x_4 must be instantiated by the same variable x_1 and the variables y_4 and y_5 must be instantiated by the same variable y_1, so that we get back $\mathcal{E}_1$.

In the structural matching principle, generalizations are computed by matching objects that compose the descriptions. For instance $\mathcal{E}_1$ and $\mathcal{E}_2$ can be generalized into:

$$(large\ x_3) \wedge (on\ y_3\ x_3) \wedge (small\ y_3) \quad (6)$$

which means that in both observations, there is a small object put on a large one, or

$$(square\ x_3) \wedge (rectangle\ y_3) \quad (7)$$

which means that in both observations there are a square and a rectangle.

The generalization given by formula 6 is obtained by matching the object represented by the variable x_1 with the object represented by x_2 and the object represented by y_1 with the object represented by y_2, while the generalization represented by the formula 7 is obtained by matching the object represented by

the variable x_1 with the object represented by y_2 and the object represented by y_1 with the object represented by x_2.

These two generalizations are not comparable from the point of view of generality. In our system, we choose the generalization that maximizes the similarity between the objects that we match. In the previous example, we choose the generalization given in 6, since it conveys more information. Weights can be put on some predicates; they enable to give information on the predicates that seem the most relevant to the problem to solve. For instance, higher weights on the predicates *rectangle* and *square* that represent the form of the objects would enable to choose the generalization 7.

In the structural matching principle, it is also possible to duplicate objects and then to get generalizations more specific than 6 and 7, but then, we have to deal with the same problems as the problems encountered with the least general generalization approach, that is to say complexity and comprehensibility problems.

The generalization function that we have implemented differs from the system OGUST [14] previously developed: It is simpler than the first one, since no background knowledge is used to detect hidden similarities between observations and when background knowledge has been used, it has been applied to the observations before the learning process. It is also more expressive than it, since constants and numeric values have been introduced and knowledge can be expressed in different ways. Moreover, it is based on the transformation of the logical representation into the more "object-oriented" one.

3.3 Similarity measure

The similarity between two logical representations is based upon the similarities of the objects that compose it. The characteristics of the objects are well expressed in the object-oriented representation. It explains why we use this representation to compute the similarity.

In order to handle numeric values, numeric predicates of arity 2 can be defined. They enable to link an object with its corresponding value for this predicate, as for instance (*age John* (30)). The specification of a numeric predicate must include its name and the length of its definition intervals, this length must be finite.

In most applications, some predicates are more relevant than other ones. In order to take this into account, weights are put on predicates. By default, they are equal to 1.

In the following, if p is a predicate, we note *weight*(p) the weight of p and if p is a numeric predicate, we note *length(p)* the length of its definition interval.

Definition 5. *Let C_1 and C_2 be two characteristics. We define*

- $sim(C_1, C_2) = 0$, *if* $C_1 = (p\ t_1 \dots t_n)$, $C_2 = (q\ s_1 \dots s_l)$ *and* $p \neq q$

- $sim(C_1, C_2) = weight(p) * \frac{length(p) - |l_1 - l_2|}{length(p)}$, *if* p *is a numeric predicate,* $C_1 = (p\ \$x\ (l_1))$ *and* $C_2 = (p\ \$x\ (l_2))$
- $sim(C_1, C_2) = weight(p)$, *if* $C_1 = C_2 = (p\ t_1 \ldots t_n)$.

Definition 6. *Let* $\mathcal{D}_1^{obj}$ *and* $\mathcal{D}_2^{obj}$ *be two object-oriented representations. Let* $(x_1, set\_char(x_1, \mathcal{D}^{obj})) \in Obj(\mathcal{D}_1^{obj})$ *and* $(x_2, set\_char(x_2, \mathcal{D}^{obj})) \in Obj(\mathcal{D}_2^{obj})$. *We define*

$$sim(x_1, x_2) = \Sigma_{C_1 \in set\_char(x_1, \mathcal{D}^{obj}), C_2 \in set\_char(x_1, \mathcal{D}^{obj})} sim(C_1, C_2)$$

We give here one of the similarity measures that we have implemented:

Definition 7. *Let* $\mathcal{D}_1^{obj}$ *and* $\mathcal{D}_2^{obj}$ *be two object-oriented representations. Let* $x_1^1, \ldots, x_{n_1}^1$ *(resp.* $x_1^2, \ldots, x_{n_2}^2$*) be the objects that appear in* $\mathcal{D}_1^{obj}$ *(resp.* $\mathcal{D}_2^{obj}$*) and let us suppose that* $n_1 \leq n_2$. *A* matching *from* $\mathcal{D}_1^{obj}$ *to* $\mathcal{D}_2^{obj}$ *is an application* M *from* $\{x_1^1, \ldots, x_{n_1}^1\}$ *to* $\{x_1^2, \ldots, x_{n_2}^2\}$.
The score of a matching M *is:* $sc(M) = \sum_{x_i^1 \in \{x_1^1, \ldots, x_{n_1}^1\}} sim(x_i^1, M(x_i^1))$.
The similarity between $\mathcal{D}_1^{obj}$ and $\mathcal{D}_2^{obj}$ *is the score of the best matching from* $\mathcal{D}_1^{obj}$ *to* $\mathcal{D}_2^{obj}$.

3.4 Example

We are currently testing this tool. To illustrate it, we give a few examples about chemical formulae. We have chosen six molecules, namely $CH3COOH$, $CH3CHO$, $CH4$, $CH2O$, $CHOOH$, $CH3OH$.

To represent the observations, we have chosen four predicates: the unary predicates *carbon*, *oxygen* and *hydrogen* describe the nature of an atom and the binary predicate *link* expresses a link between two atoms. We could have been more precise, for instance, by distinguishing simple links between two atoms from double links.

For instance, the molecule $CH4$ is represented by the list of properties[3]:
((carbon c) (link c h1) (hydrogen h1) (link c h2) (hydrogen h2) (link c h3) (hydrogen h3) (link c h4) (hydrogen h4))

No weights on the predicates: We have first run the system on two sequences of observations. In the first experiment, predicates are not weighted. The results are not very relevant, because in fact all the predicates have the same weights and all the molecules share some common properties: same kind of atoms and links. Nevertheless, it shows how the order observations are given to the system influences the result of the learning process. The only operations that have been applied are *creation* and *insertion*. The operations that enable to recover from a bad choice, that is to say *merging* and *split* have not been applied, presumably because the number of observations that have been processed is too low.

[3] The tool is implemented in a version of Scheme, therefore a conjunction of logical atoms is represented by a list.

Background knowledge: We have then introduced background knowledge that expresses functional properties of the molecule: *alcohol*, *aldehyd* and *acid* that can be expressed by theorems like:

$$\forall x \forall y \forall z \forall t \ (carbon\ x) \wedge (link\ x\ y) \wedge (hydrogen\ y) \wedge (link\ x\ z) \wedge (oxygen\ z) \Rightarrow (fn\_aldehyd\ x\ y\ z\ t)$$

We have applied the knowledge base to the observations before processing them and we have put weights only on the predicates *fn_aldehyd*, *fn_acid* and *fn_alcohol*. We obtained the following results (figure 1):

▷ Hierarchy obtained with sequence 1: $CH3COOH$, $CH4$, $CH3CH0$, $CHOOH$, $CH3OH$, $CH2O$

```
+{C0}
|-----+{ch4}
|-----+{ch3cho}
|-----+{C3}
|     |-----+{ch3cooh}
|     +-----+{chooh}
|-----+{ch3oh}
+-----+{ch2o}
```

▷ Hierarchy obtained with sequence 1 + $CH3CHO$

```
+{C0}
|-----+{ch4}
|-----+{C2}
|     |-----+{ch3cooh}
|     +-----+{chooh}
|-----+{ch3oh}
+-----+{C6}
      |-----+{ch2o}
      +-----+{C8}
            |-----+{ch3cho}
            +-----+{ch3cho}
```

Figure 1: The functions *alcohol*, *acid* and *aldehyde* have been added. Their weights are 100.

In the first column of figure 1, we can notice that the observations $CH3COOH$ and $CHOOH$ have been clustered according to their common function *acid*. $CH3OH$ which is an alcohol and $CH4$ which has no particular function remain alone. If we had to this sequence a molecule which is an aldehyde, as for instance $CH3CHO$, the molecules $CH2O$ and $CH3CHO$ are clustered according to their function *aldehyde*, as shown in the second column of figure 1.

4 Discussion and further improvements

We have presented in this paper ongoing research. We are currently experimenting it in the following directions:

Testing different similarity measures and studying their properties.

Testing other generalization functions. For the time being, we have implemented an "object-oriented" generalization that chooses the best matching between the objects that compose the descriptions. The solution that consists in considering all the possible matching has the same drawback as the least general generalization approach, that is to say, the exponential growing of the size of formulae. Syntactic biases have been introduced in [11] for clauses and we must study how they could be adapted.

Influence of the order observations are processed. The experiments that we have processed confirm a problem that is inherent in this kind of approach: the or-

der observations are presented to the system influences the result. Therefore, we would like to study on one hand, how the operations of splitting and merging enable to recover from initial misclassifications and on the other hand, the number of observations that must be given to the system before obtaining, if possible, a stable hierarchy in terms of the intensional descriptions that are learnt.
A new control strategy. Different classes that are not linked in the hierarchy may have the same intensional description. Therefore, we would like to study new operations based on the intensional descriptions of the classes and their relations of generality. Moreover, it would enable to prune the hierarchy tree.

References

1. Allen J., Thompson K. : Probabilistic Concept Formation in Relational Domains. Unknown reference.
2. Bisson G., 1992. Conceptual Clustering in a First Order Logic Representation. Proceedings of the tenth ECAI, Vienna, Austria.
3. Decaestecker C., 1991. Apprentissage en Classification Conceptuelle Incrémentale. Thèse de Docteur en Sciences, université libre de Bruxelles, 1991.
4. Fisher D.H., 1987. Knowledge Acquisition via Incremental Conceptual Clustering. Machine Learning 2, pp. 139-172.
5. Issert C., 1993. Classification Conceptuelle Incrémentale et application à la Logique d'ordre 1. Internal report, university of Orléans.
6. Kodratoff Y., Ganascia Y. 1986. Improving the generalization step in Learning, *Machine Learning: An Artificial Intelligence Approach*, Vol. 2, Michalski R.S., Carbonell J.G., Mitchell T.M. (Eds.), Morgan Kaufmann Publishers, pp. 215-244.
7. Kodratoff Y., Vrain C., 1993. Acquiring first-order knowledge about air traffic control. Knowledge Acquisition 5, Academic Press Limited, pp. 1-36.
8. Langley P., Thompson K., 1989. Incremental Concept Formation with Composite Objects. Proceedings of the Sixth International Workshop on Machine Learning, pp. 371-374.
9. Michalski R.S., 1983. A Theory and Methodology of Inductive Learning, *Artificial Intelligence*, Vol.20, N° 2.
10. Michalski R.S., Stepp R.E. 1986. Conceptual Clustering: Inventing Goal-Oriented classifications of structured objects, Artificial Intelligence 28, pp. 43-69.
11. Muggleton S., Feng C., 1992. Efficient Induction of Logic Programs. Inductive Logic programming. The A.P.I.C. Series N^o 38, S. Muggleton (Ed.), Academic Press. pp. 281-298.
12. Plotkin G.D., 1970. A note on Inductive Generalization. Machine Intelligence 5, pp. 153-163. Meltzer, Michie (Eds.), Edinburgh University Press.
13. Quinlan J.R., 1983. Learning Efficient Classification Procedures and their Application to Chess End Games. Machine Learning, an Artificial Intelligence Approach, Michalski R.S., Carbonell J.G., Mitchell T.M., eds. Tioga Publishing Company 1983, chap. 15, p. 463-482.
14. Vrain C., OGUST, a system that learns using domain properties expressed as theorems, *Machine Learning: an Artificial Intelligence Approach*, Vol. 3, pp.360-381, Morgan Kaufmann, 1990.

This article was processed using the LaTeX macro package with LLNCS style